Telecommun Syst (2012) 51:177–191
DOI 10.1007/s11235-011-9428-1

A tabu search algorithm for routing optimization in mobile ad-hoc networks

Kil-Woong Jang

Published online: 22 February 2011

Abstract In this paper, we propose a routing optimization algorithm to efficiently determine an optimal path from a source to a destination in mobile ad-hoc networks. To determine an optimal path for the nodes is important for transmitting data between nodes in densely deployed networks. In order to efficiently transmit data to its destination, the appropriate routing algorithms must be implemented in mobile ad-hoc networks. The proposed algorithm is designed by using a tabu search mechanism that is a representative meta-heuristic algorithm. The proposed tabu search algorithm carries out two neighborhood generating operations in order to determine an optimal path and minimize algorithm execution time. We compare the proposed tabu search algorithm with other meta-heuristic algorithms, which are the genetic algorithm and the simulated annealing, in terms of the routing cost and algorithm execution time. The comparison results show that the proposed tabu search algorithm outperforms the other algorithms and that it is suitable for adapting the routing optimization problem.

Keywords Routing optimization problem · Tabu search · Genetic algorithm · Simulated annealing · Mobile ad-hoc networks · Meta-heuristic algorithm

1 Introduction

There have been many recent studies on mobile ad-hoc networks due to increased user interest in ubiquitous networking. In various environments, such as territorial, underground and underwater environments, a mobile ad-hoc network is a networking technology where users can easily communicate with their destination by deploying mobile nodes in areas where there are no infrastructure networks or in areas where it is difficult to establish the network. Mobile ad-hoc networks have a lot of applications in industry, science, military and various environments. In order to provide these applications, mobile ad-hoc networks require appropriate network protocols including medium access control and routing protocols, to efficiently transmit data to their proper destinations.

The protocols of traditional wireless networks are difficult to apply to mobile ad-hoc networks because of different network factors. Unlike the nodes of traditional wireless networks, such as cellular networks, the nodes of mobile ad-hoc networks have limited bandwidth and low-power batteries. Moreover, the nodes' batteries may be constrained, and they can be difficult to recharge and replace after the nodes have been deployed. Thus, it is necessary to consider the specific factors of the networks in the process of developing the protocols for mobile ad-hoc networks.

In terms of the routing problem in mobile ad-hoc networks, if the optimal path has not been determined for transmitting data from a source to a destination, then serious problems such as high transmission delay and high energy consumption by these nodes will occur. Thus, it is certainly necessary for a routing optimization algorithm to solve this problem. Especially, in a densely deployed network, the routing optimization problem involves a classical combinatorial optimization problem. It was proven to be a NP-hard problem with computational effort growing exponentially with the number of nodes and links in the networks [1].

A routing algorithm in mobile ad-hoc networks requires a time-constraint service to determine a path from a source to a destination since the topologies of mobile ad-hoc networks are more frequently changed than those of other types

K.-W. Jang (✉)
Department of Data Information, Korea Maritime University,
1 Dongsamdong Yeongdogu, Busan 606-791, Republic of Korea
e-mail: jangkw@hhu.ac.kr

of networks. If the routing algorithm cannot provide a time-constraint service, then the nodes cannot properly transmit data to the destinations on time. Thus, in order to provide time-constraint services, the routing algorithm should find an optimal solution within a reasonable time. In order to solve this problem, most recent studies on such problems seem to focus on heuristic algorithms. Among heuristic algorithms, the exhaustive search algorithm is a general technique that consists of systematically enumerating all possible candidates for finding a global solution for NP-hard problems, and the routing optimization problem can also be solved by using this algorithm. However, finding the optimal solution by using this algorithm requires excessive computational time. To avoid numerical difficulties and reduce the computational burden, efforts have been devoted in finding high-quality solutions in a reasonable computational time by meta-heuristic optimization techniques instead of finding a global solution [2, 3]. Though meta-heuristic algorithms do not guarantee achieving the optimum solution, these algorithms will achieve an acceptable solution in a reasonable time.

In this paper, we propose a routing optimization algorithm to minimize the route cost from a source to a destination within a reasonable time in mobile ad-hoc networks. We develop the proposed algorithm by using the tabu search algorithm that is a representative meta-heuristic algorithm. In order to make a search more efficient, we propose some neighborhood generating operations for the proposed algorithm. To evaluate the proposed algorithm, we compare the proposed algorithm with other meta-heuristic algorithms, which are the genetic algorithm and the simulated annealing, under various conditions.

2 Related work

2.1 Routing protocols for mobile ad-hoc networks

In wired networks, traditional routing protocols are usually based on link state or distance vector algorithms such as the Dijkstra algorithm or the Bellman-Ford algorithm. Vasilakos et al. [4] proposed a computational intelligence approach for optimizing the routing in hierarchical ATM networks. This approach aims to efficiently allocate the network resources while ensuring the quality of service requirements for each connection. In wireless networks, possibly mobile, multi-hop network, different approaches are required. Routing protocols should be distributed, have low overhead, be self-configuring and be able to cope with frequently changing network topologies [5]. Due to this requirement, a large number of routing protocols have recently been developed for ad-hoc networks. These protocols for mobile ad-hoc networks are commonly classified [6] as either table-driven or proactive protocols, which do try to keep accurate information in their routing tables, or as on-demand protocols, which only construct routing tables when data is sent to a destination. The destination sequenced distance vector [7] and the wireless routing protocol [8] are popular examples of table-driven protocols. Dynamic source routing [9], on-demand distance vector routing and associativity-based routing [10] are representative on-demand protocols. Energy efficiency is an important consideration for mobile ad-hoc networks. For energy efficiency, it may be desirable to use not only a single path between a source and a destination but to explore multiple paths. The traditional multiple routing schemes [11–13] have also been considered in mobile ad-hoc networks. Determining the optimized path between source and destination nodes under several constrains is a very important key in many routing protocols. However, by using traditional algorithms, it is difficult to find a solution within an acceptable computation time because if there are a large number of nodes in the network, then the routing optimization problem is NP-hard.

Some routing protocols for delay tolerant networks have also been proposed to overcome frequent, long-duration connectivity disruptions. They are classified into three types: deterministic, enforced, and opportunistic approach [14, 15]. The deterministic approach can be designed when the information of network is known in advance. Jain et al. [2] proposed not only a modified Dijkstra algorithm based on knowledge oracles, but they also presented a framework for evaluating routing algorithms in delay tolerant networks. The enforced approach provides special mobile nodes to make a connection between disconnected parts of network. Zhao et al. [3] have presented a mobility-assisted approach that uses a set of mobile nodes to provide communication service in mobile ad hoc networks. Vasilakos et al. [14] proposed a routing algorithm on opportunistic approach to delay tolerant network routing. They presented the opportunistic routing design space by drawing the correspondence between the proposed delay tolerant network taxonomy and the basic opportunistic routing building blocks.

2.2 Meta-heuristic algorithms

Meta-heuristic algorithms have been devised to solve the aforementioned problems. These algorithms can find the proper solutions for the NP-hard problems in a reasonable time. The tabu search [16, 17] is a mathematical optimization method that belongs to the local search techniques. It enhances the performance of the local search method by using memory structures. The tabu lists contain attributes that are much more effective, but they raise a new problem. Typically, more than one solution is declared as tabu and some of these solutions may possess excellent qualities that may have not yet been visited. To overcome this problem, the aspiration criteria are introduced to allow the overriding tabu

state of the solution. The commonly used aspiration criterion is allowing the use of better solutions than the best currently known solutions. The genetic algorithm [18, 19] consists of powerful and broadly applicable stochastic search and optimization techniques based on the principles of evolution theory. This algorithm has received considerable attention with respect to its potential as an optimizing technique for complex problems. It has been successfully applied to industrial engineering such as a fuzzy system, which can be referred in neuro system, bio intelligent system, and ambient intelligence applications [20, 21]. Vasilakos et al. [22] proposed a fuzzy-set logic based genetic algorithms for inter-domain routing of broadband network connections with quality of service requirements in an integrated ATM and SDH networking architecture. The simulated annealing [23, 24] has been introduced by Kirkpatrick et al. and Cerny as an alternative of the local search method. It has been successfully applied to many combinatorial optimization problems. The simulated annealing algorithm is an approach to search for the optimal solution by attempting to avoid entrapment in poor local optima by allowing an occasional uphill move of the inferior solutions. This algorithm was shown to be superior in various combinatorial optimization problems. The solution representation, the energy function and several annealing parameters have to be determined to successfully apply this algorithm to various combinatorial optimization problems. Moreover, the initial solution, initial temperature and the cooling and stopping criterion have to be initialized.

3 Problem formulation

In order to describe the formulation of the routing optimization problem, we present some notations used in the proposed algorithm. The following notations will be used throughout the remainder of this paper.

Notations

N the number of nodes
L the number of links
n_i the identification of node i
l_{ij} the direct link from node i to node j
V $\{n_1, n_2, \ldots, n_N\}$
E $\{l_{12}, l_{13}, \ldots, l_{L-1,L}\}$
S source node
D destination node
I_i ith intermediate node
v_i the cost of n_i
λ_{ij} the cost of l_{ij}
Δ set of the nodes from S and D in the routing path $= \{S, I_1, I_2, \ldots, I_m, D\}$
Λ set of the links from S and D in the routing path $= \{\lambda_{SI1}, \lambda_{I1I2}, \lambda_{I2I3}, \ldots, \lambda_{Im-1Im}, \lambda_{ImD}\}$

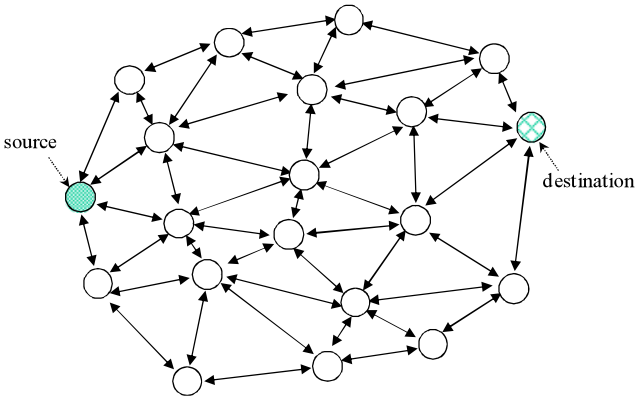


Fig. 1 Network model

C the total cost of a routing path

We first describe the network model and assumptions for the routing optimization algorithm. The network model is a flat architecture as shown in Fig. 1. The topology of mobile ad-hoc networks can be specified by a directed graph $G = (V, E)$, where V is a set of nodes with $|V| = N$ and E is a set of its links. The nodes can transmit data to another node within a transmission range. Due to the transmission range of each node, a node can directly transmit data to a destination or through another node in the network. In this figure, to transmit data from a source to a destination, each node has to send data through some intermediate nodes. The state of the links can vary according to the distance between communication nodes or the surrounding environment. Considering this point, the cost of each link is applied to a different value in the network model.

In this network model, we present some assumption to apply to the proposed algorithm. We assume every node can bi-directionally communicate with neighboring nodes via the link between the nodes. Every node has the same data processing capabilities and communication range. We search an optimal solution for the routing optimization problem that includes the total cost of nodes and links in the network model. To design the algorithm, we assume every node knows, *a priori*, the information of all the nodes and links of the networks by using another management protocol [5]. The cost of each node is determined in proportion to the number of its links, and the cost of each link is determined by the distance between source and destination nodes. Therefore, the routing optimization problem in this paper can be formulated as a combinatorial optimization problem minimizing the objective function as follows:

minimize

$$C = \sum_{i=S}^{D} v_i + \sum_{\substack{i,j=S \\ i \neq j}}^{D} \lambda_{ij} \tag{1}$$

```
X_0 = genInitSolution();
// generate stochastically an initial solution, X_0 with a uniform random number generator.
X_b = X_0, X_c = X_0; // a best solution: X_b, a current solution: X_c.
insertTabulist(X_0); // insert X_0 into tabu list.
for (i = 0; i < K_max; ) //K_max is the number of iteration.
    S_rm = removeMove(X_c);
    // generate new solutions, S_rm, by removing move for all nodes of X_c.
    S_rp = replaceMove(X_c);
    // generate new solutions, S_rp, by replacing move for all nodes of X_c.
    S = selBestSolution(S_rm, S_rp); // select a new best solution, S, among S_rm and S_rp.
    for (;;) {
        if (isTabulist(S)) // check whether S exists in tabu list.
            S = selNextSolution(S_rm, S_rp); // select a next best solution among S_rm and S_rp.
        else break; // if not, break loop.
    }
    D = cost(S) − cost(X_b); // calculate the difference value, D, between cost(S) and cost(X_b).
    if (D < 0){
        X_b = S, i = 0; // if D is negative value, assign S to X_b, and set i = 0.
    }
    else i + +; // increase the number of iteration.
    insertTabulist(S); // insert S into tabu list.
    X_c = S; // assign S to X_c.
}
B = X_b; // the final best solution, X_b, is the final solution, B.
```

Fig. 2 The procedure of the proposed tabu search

subject to

$$v_i \in \Delta \quad \text{for } i = 1, \ldots, N \tag{2}$$

and

$$\lambda_{ij} \in \Lambda \quad \text{for } i, j = 1, \ldots, L \tag{3}$$

In (1), the objective function is to minimize the routing cost of each node that is deployed in the networks. The constraint in (2) and (3) implies that we calculate the routing cost with the computation cost of nodes and the transmission cost of links between a source and a designated destination in the routing path.

4 Tabu search algorithm

The tabu search algorithm, which was first proposed by Fred Glover [16, 17], is based on using the mechanisms that are inspired by the human memory. The proposed tabu search algorithm is characterized by the following steps:

Step 1: The construction of an initial solution
Step 2: The structure of generating the neighborhood solutions

(a) The remove move operation
(b) The replace move operation
(c) Selecting an inferior solution

Step 3: Repeat step 2 until the termination criterion is met

To find an optimal solution, the tabu search first stochastically generates an initial solution with the feasible state. The initial solution becomes simultaneously a best solution and a current solution, and this solution is inserted into a memory list, which is called the tabu list. The tabu list is one of the mechanisms to prevent cycling and guide the search toward unexplored regions of the solution space. The dynamic size of a tabu list plays an important role in finding the better solutions for NP-hard problem [25]. In our experiment, for any given number of nodes N, the size of a tabu list is reset every 20 iterations to the value of between $[N, 3N]$ uniformly distributed. Once the tabu list is full, the oldest element of the tabu list is removed as a new one is added. The tabu search generates neighborhood solutions for the current solution, and the tabu search then updates the current solution with the tabu list during successive iterations. In each iteration, the set of neighbors of the current solution is built by the neighborhood generating operations, and only the neighbor with the highest value is selected as the new best solution of the next iteration. If there is no new best solution in the tabu list, then the new best solution is accepted in the process

of selecting candidate solutions; otherwise, another solution that has the next highest value becomes a candidate solution. The cost of the new best solution is compared with the cost of the best solution. If the cost of the new best solution is better than the cost of the current best solution, then the new best solution is accepted as the best solution; otherwise, the number of iteration only increases. Irrespective of the result of the cost function comparison, the new best solution is inserted into the tabu list, and it is assigned the current solution of the next generation. Figure 2 presents the procedure of the proposed tabu search for the routing optimization problem.

4.1 Solution encoding

We consider solution encoding for producing an initial solution in the meta-heuristic algorithm. Traditionally, solution encodings have been binary strings [26]. Encoding with using integer values can be more efficient for the combinatorial optimization. This was the approach that we took in this analysis. We need a table for encoding the solutions, i.e., the node table. The node table includes the identification of the neighboring nodes and link cost. According to this table, the solutions of the proposed algorithm consist of the sequences of integer values identifying the nodes that represent a path from a source to a destination, as is shown in Fig. 3.

4.2 Initial solution production

If the solution encoding is decided, then the initial solution for the proposed algorithm is made as follows. The identification of the source node is inserted into the first element of the initial solution. Next, a neighboring node that connects with the source node is randomly selected. If the selected node is not a member of the initial solution, then the selected node is inserted into the initial solution. If not, then the selected node is discarded and another node is randomly selected again. This reason is to prevent a routing loop. According to the same mechanism, a neighboring node that connects with the selected node is randomly selected and is inserted into the initial solution. The mechanism for producing the initial solution is iteratively operated until the selected node is the destination node. The initial solution produced by the production operation is calculated according to the cost function, Cost(), which calculates the cost of each solution by using the objective function. Figure 4 presents the production procedure of the initial solution for the proposed algorithm.

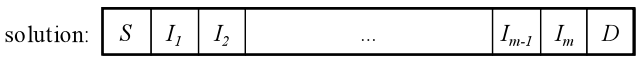


Fig. 3 Solution encoding

4.3 Repair function

Before describing the neighborhood generating operations of the proposed tabu search algorithm, we first present the repair function that is used for infeasible solutions. Meta-heuristic algorithms use several neighborhood generating operations to obtain an optimal solution. For example, the proposed tabu search algorithm uses two neighborhood generating operations for achieving the optimal solution. The solution produced from the neighborhood generating operations is divided into the feasible or infeasible solutions. In most of the previous studies, the infeasible solution is illuminated or regarded as the feasible solution by using the penalty function. However, we change the infeasible solution into the feasible solution by using a repair function.

The repair function has a mechanism that modifies an infeasible solution to a feasible solution. The repair function of the proposed algorithm is divided into two mechanisms; *Repair 1* and *Repair 2*. After performing the neighborhood generating operation of the proposed algorithm, if a new produced solution is infeasible, *Repair 1* is applied to this infeasible solution. After *Repair 1* has been carried out completely, if the solution is feasible, the solution is accepted as a candidate solution. If not, we perform *Repair 2*. After *Repair 2* is performed, if the solution is feasible, the solution is accepted as a candidate solution; otherwise, the solution is rejected. Figure 5 presents the diagram of the repair function for the proposed algorithm.

By using a network example, as is shown in Fig. 6(a), we describe the operation of the repair function. In the net-

We insert the source node into $X_0[0]$, where X_0 is an initial solution.
for $(i = 0; ;)$ {
    $s =$ selNode$(X_0[i++])$; // select randomly a node connected with $X_0[i]$, where $s \notin X_0$
    if $(s == D)$ break;
    else insNode(s, i, X_0); // insert s into X_0
}
insNode(s, i, X_0); // insert D into X_0
Cost(X_0); // calculate the cost of X_0

Fig. 4 The production procedure for the initial solution

work, we assume that node 1 is a source, and node 6 is a destination. In the first step, ***s1***, of Fig. 6(b), solution 1 is the current solution, and the second element, 2, of the solution is removed by the remove move operation of the tabu search. *Repair 1* first finds the position, called the infeasible point, of the infeasible solution. That is, *Repair 1* checks the location of the solution that is disconnected between neighboring elements of the solution. After performing the remove move operation, 1 and 5 of the solution in the second step, ***s2***, have no connection; thus, this solution is infeasible. To make this solution feasible, *Repair 1* checks elements in the solution. If any element is found, then it is inserted into the infeasible point. Using this operation, in ***s2*** of Fig. 6(b), node 3 is selected in the solution and is then inserted into the second position of the solution. During this operation, intermediate nodes are deleted in the solution. Thus, node 5 is removed in the solution in ***s3*** of Fig. 6(b). If no element is found in ***s2***, then *Repair 2* is performed for this solution. *Repair 2* also finds the infeasible point of the solution. In Fig. 6(c), solution 2 is the current solution and after performing the remove move operation of the tabu search, the infeasible point of the solution is the second position of the solution. This case cannot be feasible by *Repair 1*, and thus *Repair 2* is performed for this solution. We check whether or not there is a node that can concurrently connect with nodes 1 and 5, where the node is excluded in solution 2. Thus, node 2 is excluded and node 3 is selected. The selected node 3 is inserted between nodes 1 and 5. Finally, the solution is feasible. However, after performing two repair mechanisms, if any solution is infeasible, then the solution is rejected. Next, we describe two neighborhood generating operations of the proposed tabu search algorithm.

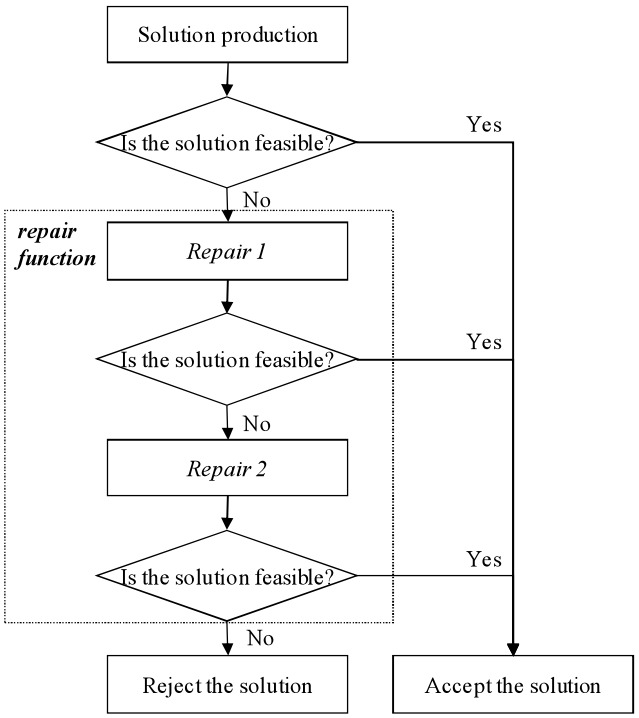
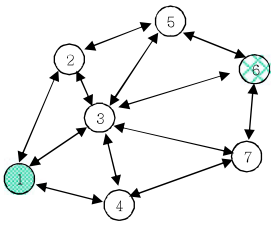
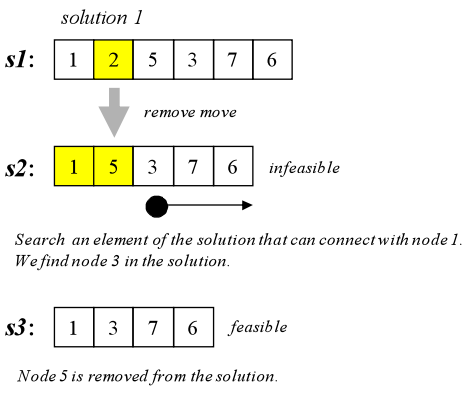


Fig. 5 A diagram of the repair function

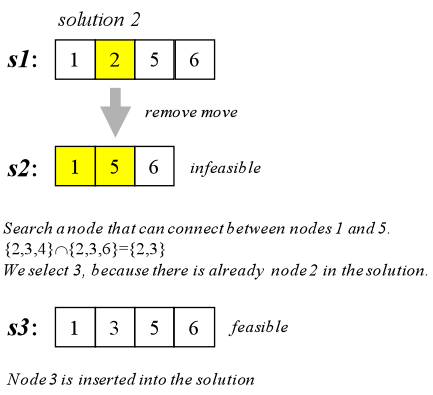
Fig. 6 An example of the repair function



(a) An example of network



(b) An example of *repair 1*



(c) An example of *repair 2*

4.4 Remove move operation

In the remove move operation, an element of the current solution is selected in the regular sequence, and then the selected element is deleted from the current solution. For example, if the second element is selected as shown in Fig. 7, then node 4 is removed from the current solution, and a new solution is produced. If the new solution is feasible, then it is accepted. However, if not, the current solution is changed to feasible by using the repair function, as is shown in Fig. 7. A new solution produced by the neighborhood generating operations is calculated according to the cost function with the objective function. Figure 8 presents the procedure of the remove move operation of the proposed tabu search.

4.5 Replace move operation

Like the remove move operation, the replace move operation is applied to all the elements of the current solution in a regular sequence. If the current node has more than an outgoing link, then a neighbor node of the current node is selected. A new solution is produced by replacing the selected node by the current node of the current solution. If the new solution is feasible, then it is accepted; otherwise, the state of the current solution is changed to feasible by using the repair function, as is shown in Fig. 9. Fig. 10 presents the procedure of the replace move operation of the proposed tabu search.

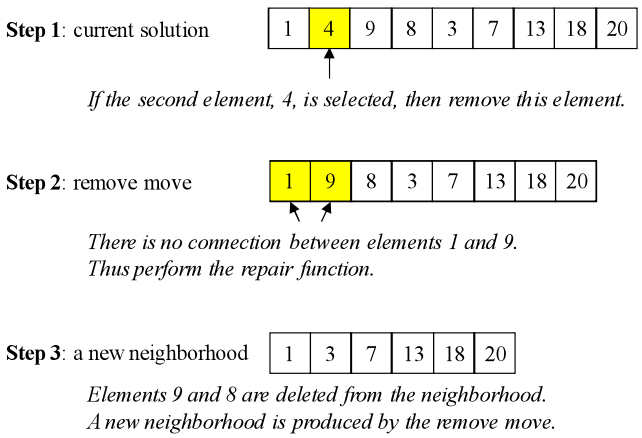


Fig. 7 An example of the remove move operation

4.6 Termination criterion

The termination criterion of the proposed tabu search algorithm is defined as the number of iterations without finding an improvement in the best feasible solution. The performance of the proposed tabu search algorithm, when the termination criterion is 10, 50, 100, or 200 iterations, is evaluated by numerical examples in Sect. 6.

5 Other meta-heuristic algorithms

In order to evaluate the proposed tabu search algorithm, we compare it with other meta-heuristic algorithms, which are the genetic algorithm and the simulated annealing, because other existing algorithms have not been addressed for the proposed network model. In addition, since the exhaustive search algorithm searches all candidate solutions, it is practically impossible to find an optimal solution for this problem using the exhaustive search algorithm. Therefore, we compare the proposed tabu search algorithm with other meta-heuristic algorithms. To do this comparison, we develop two representative meta-heuristic algorithms, which are the genetic algorithm and the simulated annealing, for the routing optimization problem. Two algorithms also use the same solution encoding, the production operation of the initial solution and the repair function proposed in this paper. We next describe the operation of the genetic algorithm and the simulated annealing.

5.1 Genetic algorithm

The genetic algorithm, which was introduced by Holland [27] and was further described by Goldberg [28], is a stochastic optimization technique. The genetic algorithm is characterized by the following steps:

Step 1: The generation of the initial population
Step 2: The selection of the parent solutions for breeding

```
for (i = 1; i < length(X_c); i++){//X_c is a current solution.
    X = removeOp(X_c, i); // generate a new solution, X, by subtracting ith node of X_c.
    // X: a new feasible solution derived by the remove move operation
    if (feasible(X)) accept(X); // if X is feasible, accept X.
    else repair(X); // if X is infeasible, call repair function.
    Cost(X); // calculate the cost of X.
}
```

Fig. 8 The procedure of the remove move operation

(a) The crossover operation
(b) The mutation operation

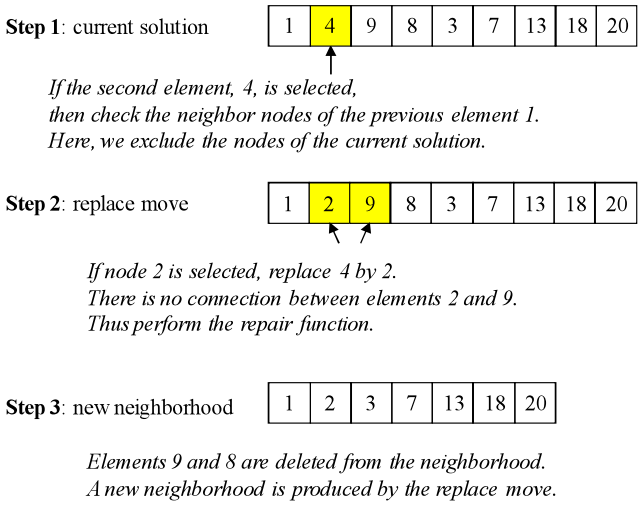


Fig. 9 An example of the replace move operation

(c) Selecting the inferior solutions

Step 3: Repeat step 2 until the termination criterion is met

The crossover and mutation operations are of significant importance for achieving an effective genetic algorithm. The crossover operation dictates the rate of convergence, while the mutation operation prevents the algorithm from prematurely converging with a local optimal solution. The numbers of solutions that are produced by each generation are tunable parameters that remain constant during a specific iteration. Figure 11 presents the detail procedure of the genetic algorithm for solving the routing optimization problem.

5.1.1 Crossover operation

The crossover operation between two chromosomes is conducted among each corresponding set of genes with the crossover probability p, as is shown in Fig. 12. First, two chromosomes are selected as the crossover partner; next,

```
for (i = 1; i < length(X_c); i++){//X_c is a current solution.
  if (degree(X_c[i]) > 1) {// if the degree of X_c is more than one
    B = selNode(X_c[i]) // select a neighbor node, B, of X_c[i].
    X = replaceOp(X_c, i, B);
    // generate a new solution, X, by replacing B with the ith node of X_c.
    // X is a new feasible solution derived by the replace move.
    if (feasible(X))accept(X); // if X is feasible, accept X.
    else repair(X); // if X is infeasible, call repair function.
    Cost(X); // calculate the cost of X.
  }
}
```

Fig. 10 The procedure of the replace move operation

```
X[N_p] = genInitChromosome();
/* generate stochastically initial chromosomes, X[N_p], with a uniform random number, where N_p is the number
   of population */
for (i = 0; i < K_max; i++) {//K_max is the generation number.
    X[N_c] = crossover(N_p, X[N_p], p);
    // generate new chromosomes, X[N_c], using crossover method with crossover probability p.
    // N_c is the number of population that are produced by the crossover method.
    X[N_m] = mutation(N_p, X[N_p], q);
    // generate new chromosomes, X[N_m], using mutation method with mutation probability q.
    // N_m is the number of population that are produced by the mutation method.
    X[N_p]_new = selNewChromosome(X[N_p], X[N_c], X[N_m]);
    // select new best chromosomes, X[N_p]_new, among X[N_p], X[N_c], and X[N_m].
    X[N_p] = X[N_p]_new; // assign X[N_p]_new to current chromosomes, X[N_p].
}
B = selBestChromosome(X[N_p]); // select the best chromosome, B.
```

Fig. 11 The procedure of the genetic algorithm

```
makePairs(X[N]); // make pairs for all the chromosomes.
// X[N] is current(initial) chromosomes and N is the population of chromosomes.
for (i = 0; i < N*p; i++) {
    // randomly select a pairs(c1, c2) in X[N], where c1 and c2 are chromosomes of X[N].
    selPairs(X[N]);
    k = rand(1, min(size(c1), size(c2))); // randomly generate a cross-point, k.
    X_c[i] = exchangePairs(c1, c2, k);
    // exchange corresponding genes of each chromosome from position k.
    // X_c[i] are two new chromosomes derived by the crossover operation.
    if (feasible(X_c[i])) accept(X_c[i]) // if X_c[i] is feasible, accept X_c[i].
    else repair(X_c[i]) // if X_c[i] is infeasible, call repair function.
    Cost(X_c[i]); // calculate the cost of X_c[i].
}
```

Fig. 12 The procedure of the crossover operation

```
for (i = 0; i < N*q; i++) {
    x = selChromosome(X[N]); // randomly select a chromosome, x.
    s = selGene(x); // randomly select a position, s, of x.
    g = genNewGene(gene(s)); // randomly generate a new gene, g.
    X_m[i] = exchange(x, s, g); // replace the gene of position s of x by g.
    // X_m[i] is a new chromosome derived by the mutation operation.
    if (feasible(X_m[i])) accept(X_m[i]); // if X_m[i] is feasible, accept X_m[i].
    else repair(X_m[i]); // if X_m[i] is infeasible, call repair function.
    Cost(X_m[i]); // calculate the cost of X_m[i].
}
```

Fig. 13 The procedure of the mutation operation

the crossover operation exchanges the corresponding genes of the two chromosomes. In the crossover operation, all the corresponding lower genes are exchanged when a gene of a chromosome is exchanged with the corresponding gene of another chromosome. The crossover procedure of the genetic algorithm is as follows:

Step 1: Group the chromosomes of the initial population by pairs and then select a pair of chromosomes.
Step 2: Randomly generate a cross-point and then exchange the corresponding genes of each chromosome from the cross-point.
Step 3: Check whether or not the two new chromosomes are feasible. If not, then change the state of the chromosome into feasible by using the repair function.
Step 4: Repeat step 2 during the iterations of (N^*p).

5.1.2 Mutation operation

The mutation operation is applied to the set of genes of all the chromosomes with the mutation probability q, as is shown in Fig. 13. The mutation operation changes or flips a gene of the candidate chromosomes to keep away from the local optima. The mutation procedure of the genetic algorithm is as follows:

Step 1: Randomly select a population of chromosomes and then select a gene of this chromosome.
Step 2: Randomly generate a new gene and then replace the selected gene of the candidate chromosome by the new gene.
Step 3: Check whether or not a new chromosome is feasible. If not, then change its state into feasible by using the repair function.
Step 4: Repeat step 2 during the iterations of (N^*q).

5.2 Simulated annealing

The idea of the simulated annealing is based on two results of statistical physics. First, if a physical system has a given energy when the thermodynamic balance is reached at a given temperature, then the probability of the system is proportional to the Boltzmann factor. Second, the Metropolis algorithm [29] can be utilized to simulate the evolution of a physical system at a given temperature.

The simulated annealing produces an initial solution by using the method of creating the initial solution that was pre-

```
X_0 = genInitSolution();
// generate stochastically an initial solution, X_0 with a uniform random number generator.
X_b = X_0; // best solution: X_b.
for (i = T_init; i < T_final; i* = α) {
// T_init is an initial temperature, T_final is a final temperature, and α is a cooling parameter.
    for (j = 0; j < K_max; j++) {//K_max is the number of iteration.
        for (k = 0; k <length(X_b); ) {
            if (rand(0, 1) < β) // β is an operation threshold.
                X = removeOp(X_b); // generate a new solution, X, by the exchange operation.
            else
                X = replaceOp(X_b); // generate a new solution, X, by the replace operation.
            D = cost(X) − cost(X_b);
            // calculate the difference value, D, between cost(X) and cost(X_b).
            if (D < 0)X_b = X; // if D is a negative value, assign X to X_b.
            else if (rand(0, 1) < exp(D/i))X_b = X;
            // by the Boltzmann distribution for energy states at thermal equilibrium.
            else k++;
        }
    }
}
B = X_b; // the final best solution, X_b, is the best solution, B.
```

Fig. 14 The procedure of the simulated annealing

viously mentioned. The algorithm decreases a given temperature by multiplying the cooling parameter, α, of the initial temperature, T_{init}, by the final temperature, T_{final}. During this period, the algorithm is iteratively operated. In the algorithm iteration, new solutions, X, are produced by one of the two neighborhood generating operations that adapt to the current solution, X_b. The two neighborhood generating operations are the exchange and the replace operations that have been described in the tabu search. The probability of selecting the neighborhood generating operations depends on the given operation threshold, β. The cost of X is compared with the cost of X_b. According to the comparison result, X is accepted as X_b, or X is applied to the Boltzmann factor. That is, if the difference value, D, between the cost of X and the cost of X_b is less than zero, then X is accepted as X_b; otherwise, a random number that is uniformly distributed in the interval (0, 1) is selected, and this number is compared with the Boltzmann factor, $\exp(D/T)$. Here, T is a control parameter that is known as the system parameter. If this number is less than the Boltzmann factor, then X is accepted as X_b; otherwise, X is rejected and X_b is kept. Figure 14 presents the overall operation of the simulated annealing for the routing optimization problem.

6 Numerical examples

In this section, we compare the proposed tabu search algorithm with two meta-heuristic algorithms, the genetic algorithm and the simulated annealing, via computer experiments. All the experiments are implemented with the use of C++, and they are performed on a 1.80 GHz Pentium®4 that is equipped with a Windows OS, 2 GB of memory and an Intel® processor. The algorithms were applied to optimize the routing problems with four different network topologies. The topologies are called problems A, B, C and D. Each problem contains some nodes and links, as is shown in Table 1. Moreover, the parameters of the algorithms that are used in the experiments are shown in Table 2.

Table 1 Problems for the experiment

Problem	# of nodes	# of links
A	80	225
B	160	456
C	320	923
D	640	1862

We measured the routing cost of the tabu search with the number of iterations: 10, 50, 100 and 200. Figure 15(a) plots the minimum routing cost as a function of four problems for the proposed tabu search. In general, if the number of iterations increases in the tabu search algorithm, the probability of finding the optimal solution increases. In this figure, we observe that the results of the minimum routing cost are similarly represented irrespective of the number of iterations in the small size network. This means that the proposed algo-

rithm can find an optimal solution in the small size network though a small number of iterations are applied to the proposed tabu search. On the other hand, by increasing the number of nodes in the network, we see that the tabu search with the larger number of iterations finds an optimal solution with better performance. However, the minimum routing cost of the tabu search with $K_{max} = 100$ is similar to that of the tabu search with $K_{max} = 200$. This is due to the fact that there is little gain in exceeding $K_{max} = 100$ for this problem. Figure 15(b) shows the average execution time as a function of four problems of the proposed tabu search. For all problems, by increasing the number of iterations, the average execution time of the tabu search increases. The average execution time and the routing cost of the nodes are trade-off factors, so these two factors must be simultaneously considered and balanced by the routing algorithm.

Table 2 The parameters of the meta-heuristic algorithms

Algorithms	Parameters	Values
Tabu search	K_{max}	10/50/100/200
Genetic algorithm	N_p	50
	p	1/0.5
	q	1/0.5/0.25
	K_{max}	100
Simulated annealing	T_{init}	0.1
	T_{final}	0.00005
	α	0.1
	β	0.1/0.3/0.5/0.7/0.9
	K_{max}	100

The crossover probability, p, is set to 1 or 0.5, and the mutation probability, q, is set to 1, 0.5 or 0.25, in all the genetic algorithm experiments. We run the experiment under the combinations of two parameters, p and q. We experimented with six cases of mixed crossover probability and mutation probability to measure the optimal cost of solution of the genetic algorithm; if the value of p is equal to 1, the crossover operation for all the population of the genetic algorithm is carried out; however, if the value of p is equal to 0.5, then the crossover operation for half the population is carried out. Likewise, the mutation operation is performed according to the value of q. The initial population for problems A, B, C, and D is set to 50, and the generation number is set to 100. We chose the best case of the cost and the execution time for 10 runs. Figure 16(a) shows the minimum routing cost as a function of four problems for the genetic algorithm. The results of minimum routing cost for all the cases are slightly different in relation to p and q in all the problems. Especially, the performance of the generic algorithm with a high mutation probability is more improved in other cases. Figure 16(b) shows the average execution time as a function of four problems of the genetic algorithm. This figure shows that by increasing p and q, the average execution time of the genetic algorithm increases. Moreover, the results of average execution time for all the cases also increase in proportion to the number of nodes in

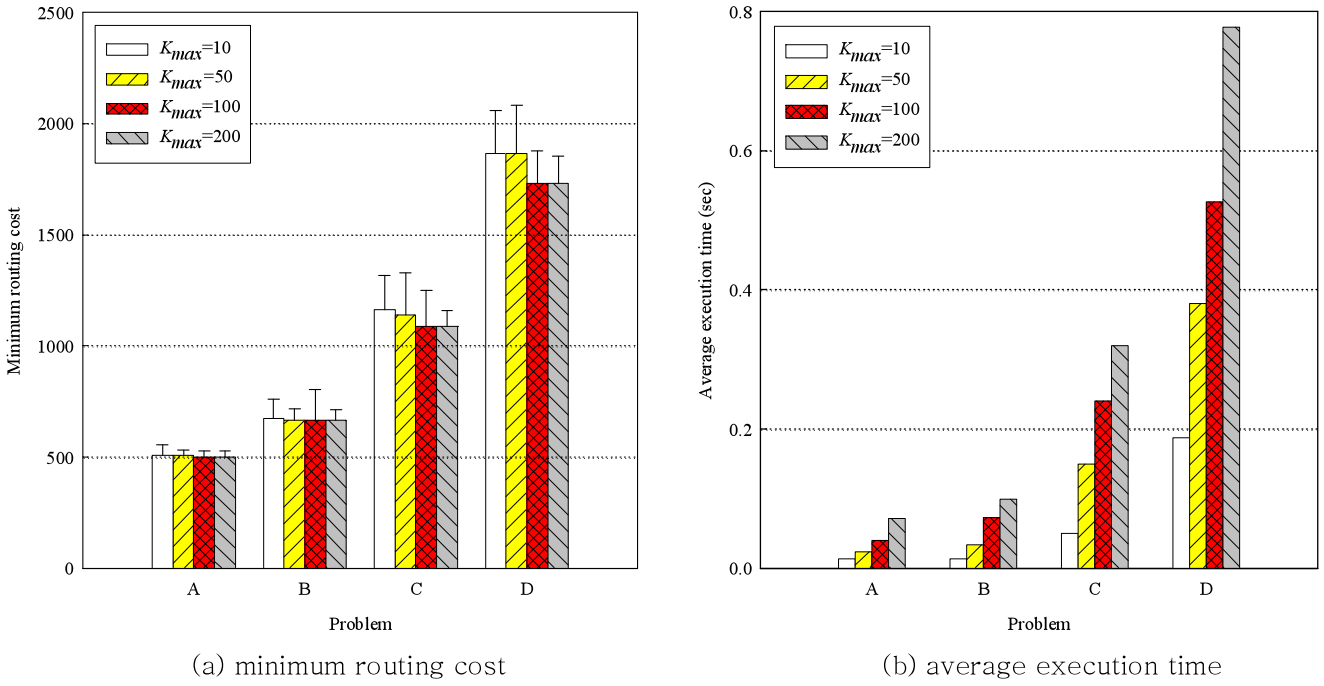
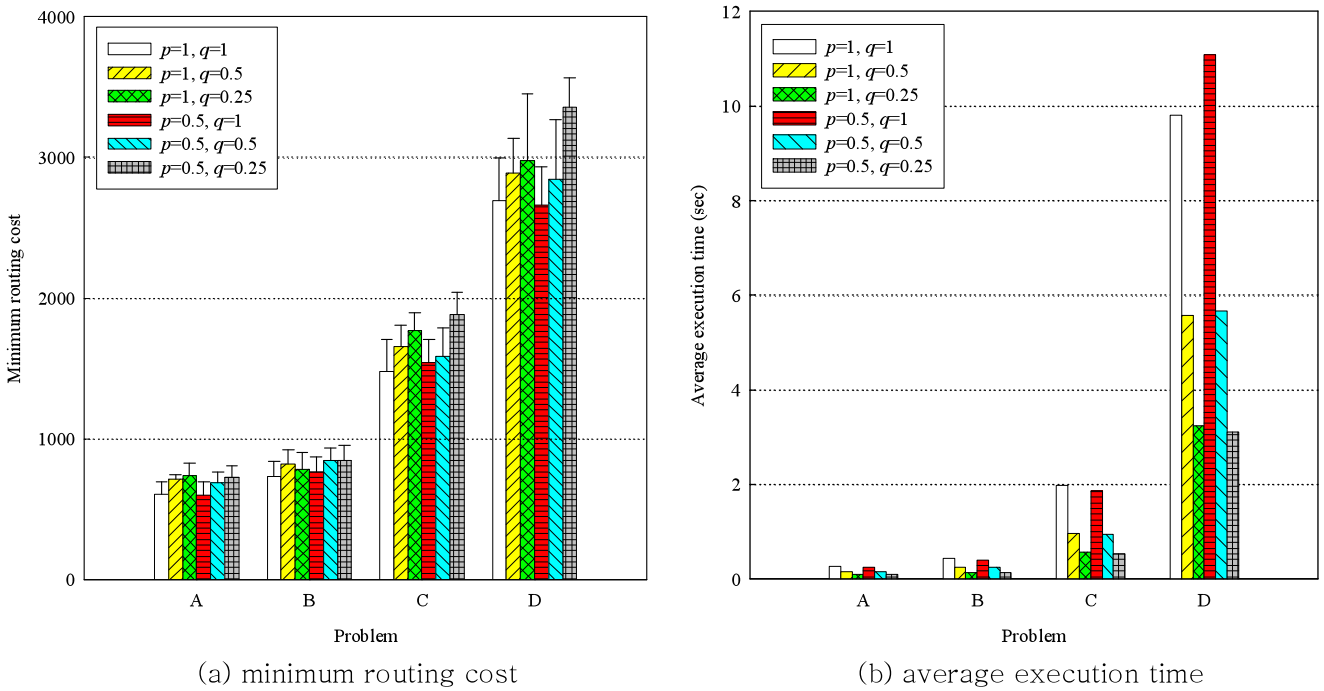
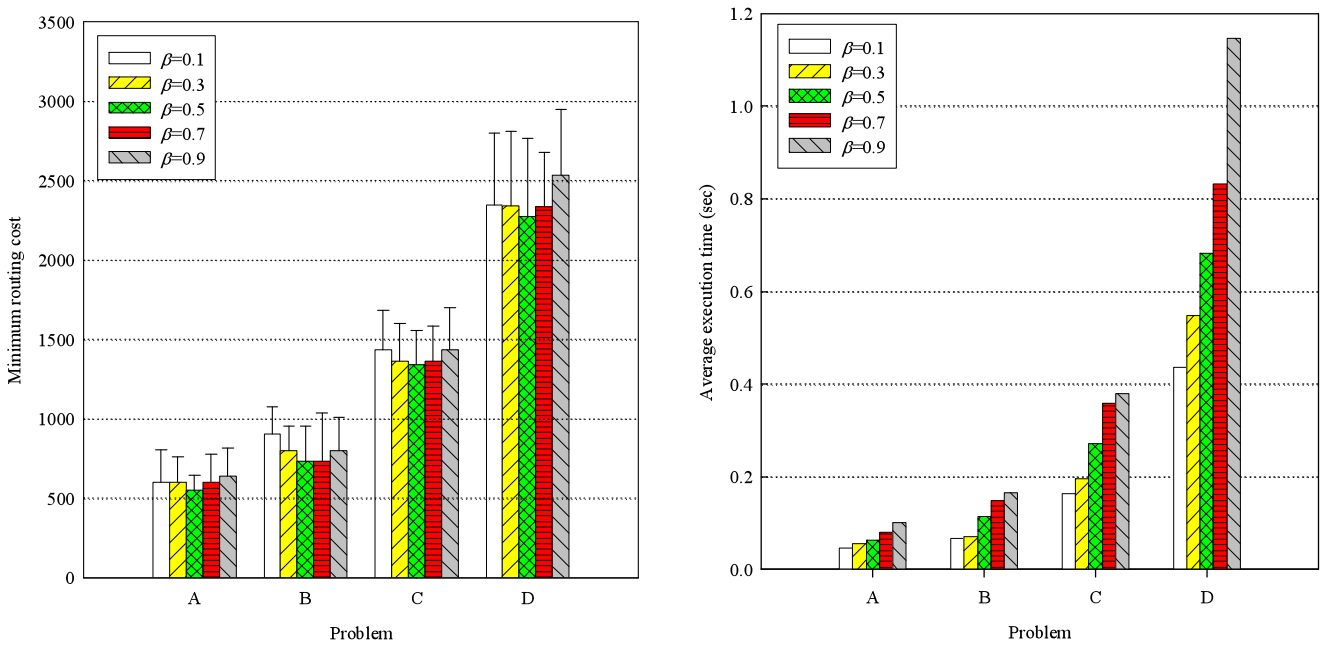


Fig. 15 Comparison results as a function of four problems for the tabu search



(a) minimum routing cost (b) average execution time

Fig. 16 Comparison results as a function of four problems for the genetic algorithm



(a) minimum routing cost (b) average execution time

Fig. 17 Comparison results as a function of four problems for the simulated annealing

the four problems. In Fig. 16(a) and (b), we can see that we must also consider simultaneously the probabilities of the crossover and mutation to minimize the average execution time and the routing cost of nodes for solving the routing problem with using the genetic algorithm.

Figure 17(a) shows the minimum routing cost as a function of four problems for the simulated annealing. We measured the routing cost of all the problems for five operation thresholds, $\beta = 0.1$, 0.3, 0.5, 0.7 and 0.9. The simulated annealing randomly changes the element of an intermedi-

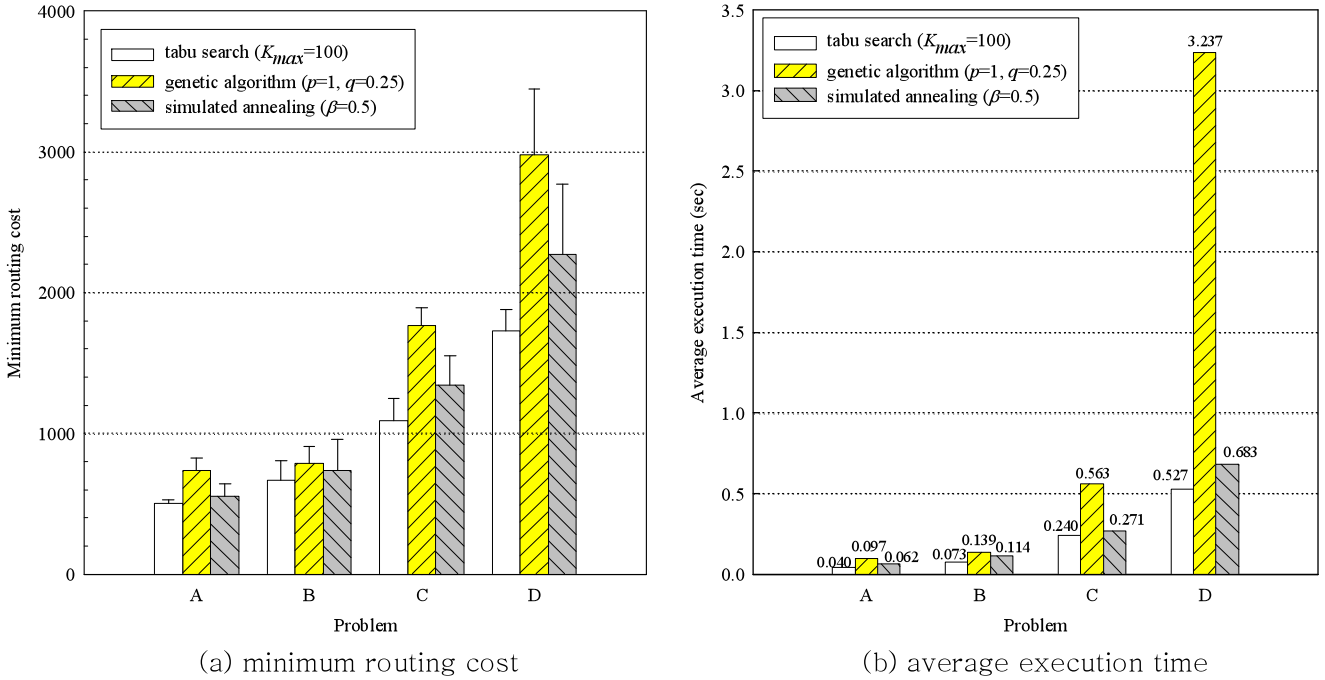


Fig. 18 Comparison results as a function of four problems for each algorithm

ate solution by using the remove and replace operations. If β is equal to 0.3, then the remove operation is operated as much as 30% in total neighborhood generating operations, and the replace operation is operated as much as 70%. By increasing the network size, each case has a different result. In $\beta = 0.5$, the simulated annealing has a slightly better result than those of other cases. This is because the two neighborhood generating operations are well applied to this problem.

Figure 17(b) shows the average execution time as a function of four problems for the simulated annealing. As previously mentioned, we can see that by increasing the network size, the average execution time of the simulated annealing also increases. In the experiment of the simulated annealing, we observed that solutions produced from the remove operation are more infeasible than ones produced from the replace operation. If the solutions are infeasible, we perform the repair function to change the infeasible solutions to feasible. Therefore, as β increases, the average execution time of the simulated annealing will takes a longer time.

Figure 18(a) shows the comparisons of the minimum routing cost of three meta-heuristic algorithms: tabu search, genetic algorithm and simulated annealing. We applied three algorithms to the given problems according to the following conditions. In the tabu search, K_{max} is set to 100. In the genetic algorithm, p and q are set to 1 and 0.25, respectively, and in the simulated annealing, β is set to 0.5. We see that in terms of the routing cost, the tabu search algorithm outperforms the genetic algorithm and the simulated annealing. The tabu search and the simulated annealing algorithms exploit the same neighborhood generating operations, which are the remove and the replace operations, but since the tabu search regularly produces more next generation solutions than the simulated annealing, we observe that the tabu search has better performance than the simulated annealing in the figure. We also see that the tabu search outperforms the genetic algorithm with the crossover and the mutation operations. This main reason is that the tabu search with regularly generating neighborhood solutions is better applied to the routing problem than the genetic algorithm with stochastically generating solutions. Figure 18(b) shows comparisons of the average execution time for the meta-heuristic algorithms in the given problems. The results show that the tabu search takes a shorter time than the genetic algorithm and the simulated annealing. This is because the tabu search reaches the optimal solution faster than other algorithms. Moreover, though the tabu search produces more neighborhood solutions for the current solution than the simulated annealing, it takes a shorter time than the simulated annealing that randomly produces a next generation for the current solution.

Finally, for the routing problem in the mobile ad-hoc networks, we observe that the proposed tabu search algorithm can efficiently solve this problem in terms of the routing cost, and it is pertinent to resolve the problem within a reasonable execution time.

7 Conclusions

In this paper, we proposed a routing optimization algorithm to efficiently determine an optimal path from a source to a destination in mobile ad-hoc networks. The proposed algorithm was designed by using the tabu search algorithm, which is a typical meta-heuristic algorithm. We first described the commonly used operations, which are encoding solution, the production of the initial solution and the repair function, and we presented the neighborhood generating operations of the algorithm that includes the termination condition. To compare the performance of the proposed algorithm, we also developed the genetic algorithm and the simulated annealing that are the representative meta-heuristic algorithms. We evaluated the performance of the algorithms by carrying out the experiments by varying the number of nodes and links, and we compared the proposed algorithm with other meta-heuristic algorithms in terms of the routing cost and the average execution time for the routing problem. The comparison results showed that the tabu search outperforms other algorithms in terms of the routing cost and average execution time under various constraints, and it is suitable for adapting the routing optimization problem.

References

1. Ali, M. K., & Kamoun, F. (1993). Neural networks for shortest path computation and routing in computer networks. *IEEE Transaction Neural Networks*, *4*, 941–954.
2. Jain, S., Fall, K., & Patra, R. (2004). Routing in a delay tolerant network. In *Proceedings of ACM SIGCOM* (pp. 187–198).
3. Zhao, W., Ammar, M., & Zegura, E. (2004). A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of ACM/IEEE MOBIHOC*.
4. Vasilakos, A., Saltouros, M., Atlasis, A., & Pedrycz, W. (2003). Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques. *IEEE-Transactions on Systems, Man and Cybernetics, Part C*, *33*(3), 297–312.
5. Akkaya, K., & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, *3*, 325–349.
6. Royer, E. M., & Toh, C. K. (1999). A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, *6*, 46–55.
7. Perkins, C. E., & Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer. In *Proceedings of the ACM SIGCOMM* (pp. 234–244).
8. Raju, J., & Garcia-Luna-Aceves, J. J. (2000). A comparison of on-demand and table driven routing for ad-hoc wireless networks. In *Proceedings of ICC*.
9. Maltz, D. (2001). *On-demand routing in multi-hop wireless ad hoc networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.
10. Perkins, C. E., & Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE workshop on mobile computing systems and applications* (pp. 90–100).
11. Toh, C. K. (1999). A novel distributed routing protocol to support ad hoc mobile computing. In *Proceedings of IEE 15th annual international conference on computers and communications* (pp. 460–486).
12. Nasipuri, A., & Das, S. R. (1999). On-demand multipath routing for mobile ad hoc networks. In *Proceedings of the 8th ICCCN*.
13. Valera, A., Seah, W. K., & Rao, S. V. (2003). Cooperative packet caching and shortest multipath routing in mobile ad hoc networks. In *Proceedings of IEEE INFOCOM*.
14. Spyropoulos, T., Rais, R. N. B., Turletti, T., Obraczka, K., & Vasilakos, A. (2010). Routing for disruption tolerant networks: taxonomy and design. In *Wireless networks* (pp. 1–22).
15. Pedrycz, W., & Vasilakos, A. (2001). *Computational intelligence in telecommunications networks*. USA: CRC Press
16. Glover, F. (1989). Tabu search, Part I. *ORsimulated Annealing Journal on Computing*, *1*, 190–206.
17. Glover, F. (1990). Tabu search, Part II. *ORsimulated Annealing Journal on Computing*, *2*, 4–32.
18. Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: Univ. of Michigan Press.
19. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization & machine learning*. Reading: Addison–Wesley.
20. Pedrycz, W., & Vasilakos, A. V. (1999). Linguistic models and linguistic modeling. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, *29*(6), 745–757.
21. Acampora, G., Gaeta, M., Loia, V., & Vasilakos, A. V. (2010). Interoperable and adaptive fuzzy services for ambient intelligence applications. *ACM Transactions on Autonomous and Adaptive Systems, 5*(2).
22. Vasiliakos, A., Ricudis, C., Anagnostakis, K., Pedrycz, W., & Pitsillides, A. (1998). Evolutionary-fuzzy prediction for strategic QoS routing in broadband networks. In *Proceedings of IEEE-FUZZ* (pp. 1488–1493).
23. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.
24. Rutenbar, R. (1989). Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*, *5*, 19–26.
25. Kulturel-Konak, S., Norman, A. E., & Coit, D. W. (2003). Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions*, *35*, 515–526.
26. Antonisse, J. (1989). A new interpretation of schema notation that overturns the binary encoding constraint. In *Proceedings of the 3rd international conference on genetic algorithms* (pp. 86–91).
27. Ahn, C. W., Ramakrishna, R. S., Kang, C. G., & Choi, I. C. (2001). Shortest path routing algorithm using Hopfield neural network. *Electronics Letter*, *37*(19), 1176–1178.
28. Munemoto, M., Takai, Y., & Sato, Y. (1998). A migration scheme for the genetic adaptive routing algorithm. In *Proceedings IEEE international conference systems* (pp. 2774–2779).
29. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., et al. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 10–16.

Kil-Woong Jang received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Kyungpook National University, Korea in 1997, 1999 and 2002, respectively. Since 2003, he has been an associate professor of Department of Data Information, Korea Maritime University, Korea. His research interests include wireless network protocol and optimizations for mobile ad-hoc network.